eWAN: grid emulator

Magí Sanchón Soler

June 6, 2006

Contents

1 Preamble **3**
- 1.1 Versions . . . 3
- 1.2 Thanks . . . 5

2 Installation **8**
- 2.1 Installation program user manual (grid5000) . . . 8
  - 2.1.1 Grid5000 structure . . . 8
  - 2.1.2 How the deployment tool works . . . 10
  - 2.1.3 File struct . . . 10
  - 2.1.4 The code parts . . . 10
  - 2.1.5 Configuration variables . . . 14
- 2.2 Installation program (general purpose) . . . 18

3 Execution **20**
- 3.1 eWAN graphical interface user manual . . . 20
  - 3.1.1 The pages . . . 20
  - 3.1.2 The page's links . . . 29
  - 3.1.3 Graphical representation . . . 30
  - 3.1.4 Configuration file . . . 30

4 Post work **34**

5 Creating Images for eWAN **36**
- 5.1 Creating the images . . . 36
  - 5.1.1 The eWAN server image . . . 36
  - 5.1.2 The eWAN client image . . . 37
- 5.2 After installing the tools . . . 38
  - 5.2.1 Creating the image . . . 38
  - 5.2.2 Registering in grid5000 (Kadeploy tool) . . . 38
- 5.3 Image creation examples (on grid5000) . . . 39
  - 5.3.1 Script used to create the server image . . . 39
  - 5.3.2 script used to create the client image . . . 40

6 eWAN SSH keys (grid5000) **41**
- 6.1 List of files . . . 41
- 6.2 Functions . . . 41
  - 6.2.1 id_dsa . . . 41
  - 6.2.2 id_dsa_eWAN . . . 41

6.2.3 id_dsa_eWAN_server 42

7 Defining a topology to deploy 43
7.1 The different parts 43
7.1.1 RDF/XML definition 43
7.1.2 Topology 43
7.1.3 Node group 44
7.1.4 Client 44
7.1.5 Router 44
7.1.6 Access point 44
7.1.7 Cross traffic generator 44
7.1.8 Link group 45
7.1.9 Link 45
7.2 Complete little example 45

8 eWAN archive structure 48

9 Complete example 52
9.1 creating the eWAN images 52
9.2 Prepare the environment 54
9.2.1 The server description 55
9.2.2 The client description 55
9.3 Installation 56
9.3.1 The configuration file 56
9.3.2 Start execution 57
9.4 Execution 59
9.5 Post work 59

Chapter 1

Preamble

The eWAN objective is to allow the users to test their own applications or protocols on an emulated grid before the test or deployments on a real grid environment.

The researcher has three possibilities to test his work:

- Real grid environments: Too susceptible to real aleatory conditions, not easy to control and difficult to get a good grid account.
- Simulation option: On the other hand we have the simulation option. On simulations the scalability are maximal, but the performance of each simulated node (router, client, link) falls down by the execution of several nodes on one physical machine.
- Emulation option: The emulation doesn't scale like the simulation, since one portion is simulated and the rest is real. On each physical machine only one node is assigned, without performance limits or loss. It's an intermediate version.

EWAN is an emulator. It is deployed on a cluster, assigning each topology node to deploy to a physical node. The topological links to emulate are assigned to other machines, with a maximum of two links (in the case of the two links which connect two nodes -a link and its inversed-) for each machine.

This document offers detailed explanations over eWAN, the manner in which it works and the different ways in which you can use it. It describes the configuration options and also gives utilisation examples in order to help you to work with eWAN.

1.1 Versions

The actual version is 2.0. The previous version, 1.0, implemented by *Cyril Otal*, had not the capacity to use virtual interfaces and it only disposed of a graphical part. Version 2.0 was developed by *Magí Sanchón Soler* who has corrected and improved the initial one.

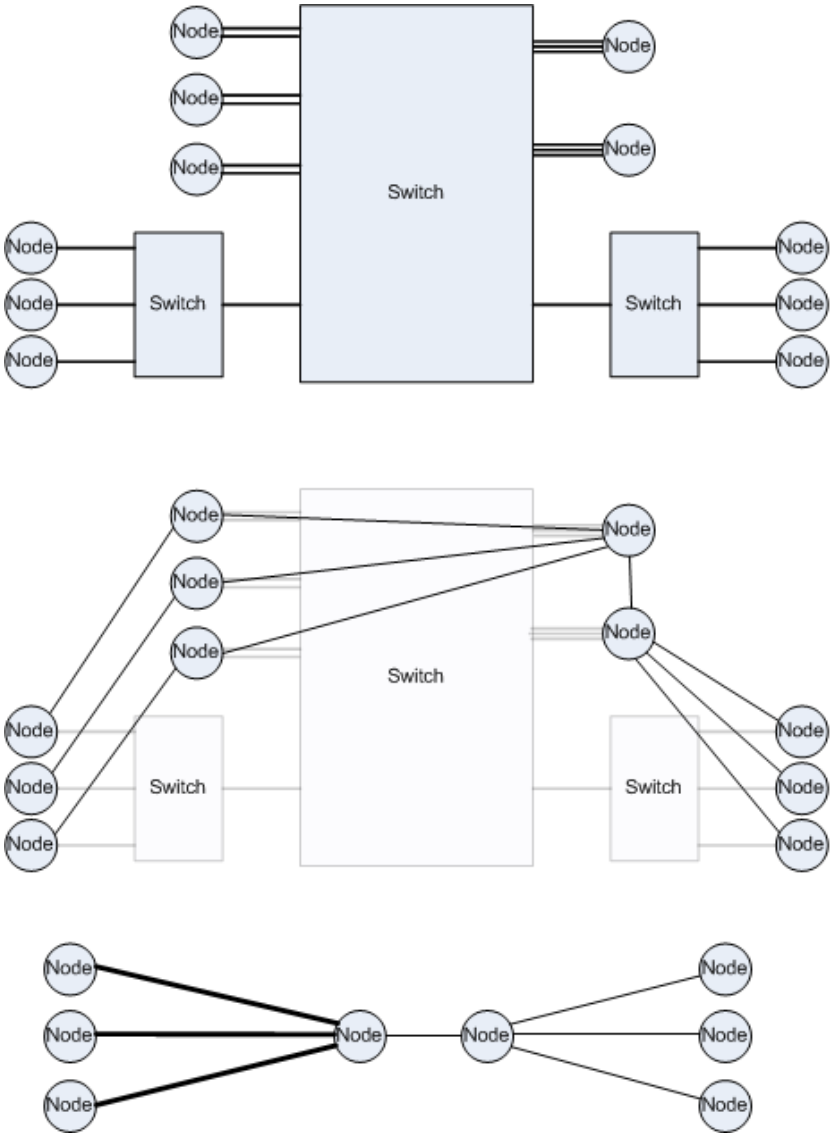

Figure 1: Transform your cluster into a grid

1.2 Thanks

Thanks to:

Raluca Uricaru

Stephan Bayer

Anna Artigas Miquel

Paolo Campiglio

and the "reso" team at ENS Lyon, especially:

Sébastien Soudan

Ludovic Hablot

Bin Bin Chen

EWAN design

The work with eWAN covers three domains: the installation, the execution and the post work. For each part eWAN gives a tool or facilities to the final user to allow easy work. Next sections define each of the three parts, and show the tools for each part with their user manuals and configuration options.

Chapter 2

Installation

The installation program allows you to install the eWAN graphical interface to finally deploy a logical topology on the cluster. It reserves, deploys and reads the configuration options of the nodes to generate the configuration file for eWAN, installs it and starts the initial eWAN graphical interface page.

If you wish, you can make the installation for yourself. The graphical interface of eWAN allows you to have some configuration files with different options on each one (the nodes, the IP's, ...). You have also a webtool to verify that the configuration file is wellwrited.

The installation script can be viewed as two tools, a cluster specific tool and a cluster discovery tool. Each cluster works with a set of tools which are different from another cluster, the security systems are not equal, and all the servers, clients, hostnames and structure are specific for each cluster. Actually, only a grid5000 structure is allowed, with the OAR reservation tool and the Kadeploy deploying tool, but the cluster discover tool can be reused on other installation programs.

2.1 Installation program user manual (grid5000)

2.1.1 Grid5000 structure

In grid5000, we have found nine sites with one or more clusters in each one. Each site has an access machine (user@acces.site.grid5000.fr) and its own user space. Unfortunately, not all sites accept our image (each cluster has its machine type, and we must make an image for each machine type) and not all sites accept a direct connection to the "access" machine with our user. Furthermore, the user space (/home/yoursite/youruser/) is not shared between sites and not all sites have the same service structure (oar, kadeploy, etc can be served in one or more machines, for example). Figure 2.1.1.

To avoid these difficulties, the eWAN deployment tool is connected to a site and can deploy another one. The images are read from each site when it is necessary (with kadeploy tool). An example on the figure 2.1.2.

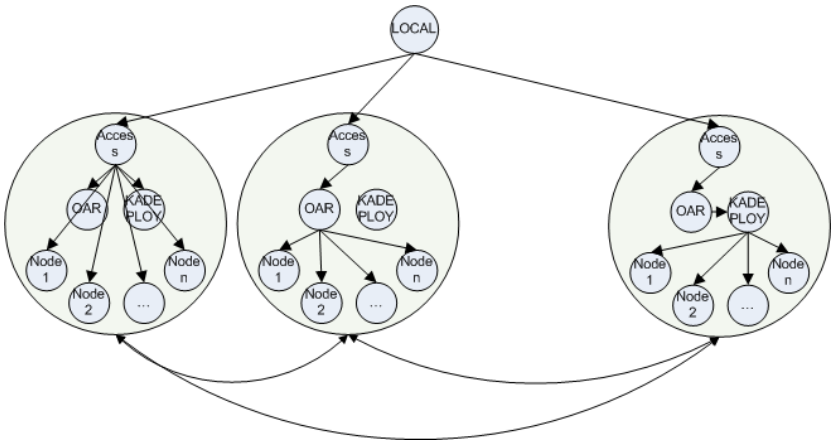

Figure 2.1.1: Grid5000 schema

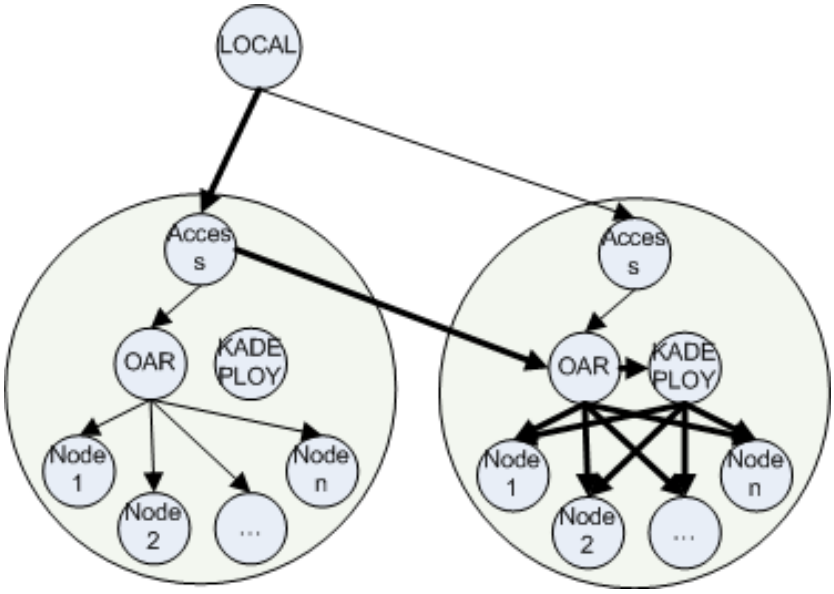

Figure 2.1.2: Deployer normal path

2.1.2 How the deployment tool works

It can't work as a normal script because of the ssh connections. It executes four phases in each site:

1. When it connects to a site, it executes the code part that must be executed on the site,
2. Then it seeks the next place where to go and if it is ready,
3. It copies the needed data (config files, code files and others) to the site
4. And it calls the next code part on the next site using a SSH command.

2.1.3 File struct

The name of all the files starts with "deployeWAN". Only one configuration file exists (named deployeWANdata). It contains all the options for each script part. Each code part calls the next one in the correct server. The sequence is:

deployeWAN↦Pre-local↦Local↦Portal↦OAR↦Kadeploy↦Server

2.1.4 The code parts

In this section I will explain the functionality of each part and their possible configurations.

Pre-local part

files:

1. deployeWAN

data:

no configuration data is used.

preconditions:

no preconditions.

We need a ssh-agent to manage the ssh-keys. In this way we can have acces to the local server and to the deployed nodes (each eWAN image contains a ssh-key to allow easy acces, without password (6)). This part tries to find an ssh-agent and if it is not already open, it does automatically.

Local part

files:

1. deployeWANlocal

preconditions:

1. the private user ssh-key must be in the following folder $HOME/.ssh/

data:

1. eWAN ssh-key
2. grid5000 user
3. portal site
4. development site

Adding the user ssh private key and the eWAN ssh private key, it executes the portal part, and, if the eWAN server is correctly deployed, it opens a tunnel connection to the server and a web browser (search with firefox or mozilla-firefox). If any browser is opened, you can open one yourself and write the indicated web address.

Portal part

files:

1. deployeWANportal

preconditions:

no preconditions.

data:

1. development site (given by the Local part on the call)

Look if the script is on the right site. If it is, check if it is also in the OAR server (not every server has the oar reservation tool). If is not in the oar server or if it's not on the correct site, it will create a connection with the oar.deploysite server to go to the correct oar and site servers.

OAR part

files:

1. deployeWANoar

preconditions:

no preconditions.

data:

1. usereservation
2. nodes
3. time
4. delay

Perform a reservation with the OAR tool (oarsub ...). If the usereservation is not 0, the script will not do the reservation and it will use the "usereservation" reservation instead. If any reservation is specified, the script will reserve the specified nodes for a specified duration, and it can reserve and wait to start the deployment after the specified delay. The eWAN server will be deployed (no error is allowed, if it appears, the script try on another node), but the other nodes are deployed only once, and the ones producing errors will not be taken into account later on the cluster discover tool. After doing (or using) the reservation, the script takes the nodes and puts them on a file for kadeploy. Finally, it executes the kadeploy script part.

Kadeploy part

files:

1. deployeWANkadeploy

preconditions:

1. the nodes must be specified in a file which must be a script parameter.

data:

1. serverimgname
2. clientimgname
3. serverimgfile
4. clientimgfile
5. imagepath
6. partition
7. ewansourcedeploy

Look if the images, server and eWAN client, are registered to use the kadeploy tool. If not, it registers the images from the configuration data (serverimgfile, clientimgfile, imagepath). Then, it will deploy an eWAN server. To do this, the script will take the first node and it will deploy the server image. If it does not work, it will take the next node and it will try to do the same. When the server is deployed, the rest of the reserved nodes are deployed with the client image. The nodes which can produce an error on the deployment part are removed from the list of available nodes (nodes to use in eWAN). Finally, the server executes two new parts which are optional and used as tools: the eWAN files and the configuration files.

copy eWAN

files:

1. deployeWANcopysource
2. deployeWANcopysource2

preconditions:

1. the specified eWAN source file must be in the correct format (.tar.bz2) and in the same folder.
2. the second part of the script must be in the same folder

data:

no configuration data is used.

The first part takes the data from argv (file user and node) and it makes copies of all the needed files (code, DBconfiguration and second part) to the objective node and executes the script second part. The second part starts in the eWAN server node, it unpacks the source and puts it in the correct folder. It also prepares the environment (permisions, ssh-key) and it configures the DB.

read nodes configuration

files:

1. deployeWANgetdata
2. deployeWANgetdata2

preconditions:

1. the nodes that are being read, must be in a file named "clients"
2. the correct ssh-keys must allow the server-client connections

data:

1. sshtimeout
2. cpurout
3. cpupt_acc
4. cpucli
5. tocken_buffer
6. switch_routeur
7. subnetnumber
8. subnetnumberXip
9. subnetnumberXmask

These files make the config.php file, which contains all the cluster and eWAN configuration. The network control options are set with the eWAN server node options (gateway, network and mask) and a second script is executed on each client node to get client information (IP, processor speed, network cards number and names and the network control card). The rest of the information is read using the eWAN options (listed in "data") to put them on the config.php file.

2.1.5 Configuration variables

The file containing all the deploy tool variables is divided into four parts. The comments are preceeded with a # and all the variables must be defined in the format:

variableNAME=variableVALUE

without spaces between the variable or value and the equality sign.

Now each part is explained. For each variable you have a little description of the variable name, the accepted format values, and an example.

Local data

All the data used for the connection to grid5000.

sshkey The eWAN deployment tool needs a private key to have the permission to enter on each node deployed. This key is on a local directory, and the deployment tool needs to know the path to add the key on the ssh-agent.
format:

The path must be the complete linux path (file included), with / separators.
example:

sshkey=ṡsh/id_dsa_eWAN

user To connect to the grid5000 grid, a username must be used
format:

Normal username.
example:

user=msanchon

portal and deploy sites On a grid5000 grid, the deployment tool allow you to connect to a site and deploy on another one. Portal is the site name to access, and deploysite is the site to do de reservation and deployment.
format:

Normal sitename
example:

portal=lyon

format:

Normal sitename
example:

deploysite=bordeaux

OAR data

The data used for the OAR tool or used by the installation program to do the reservation options.

user reservation If we want install eWAN on a previous reservation, we can specify the reservation number. If we want to do a new reservation, a 0 is required.
format:
Only positive numbers. With a 0 the installation will do a new reservation with the nodes time and delay variables. If no 0 specified, must be an existent JOBid number.
example:

usereservation=16185

nodes If we do a reservation, the number of nodes to reserve. One node will be used to deploy de eWAN server (will not be available to deploy topologies).
format:
A positive number.
example:

nodes=36

time If we do a reservation, we can specify the duration in hours
format:
Positive number (duration)
example:

time=8

delay If we do a reservation, we can choose to do a reservation now for later.
format:
A + sign, a space and the hours. If + 0, no delay will be applied.
example:

delay=+ 3

Kadeploy data

All the Kadeploy options, images, ...

image names We must specify the name of the images on the kaenvironment tool.
format:
Imagename
example:

serverimgname=debian4eWANserver

format:
Imagename
example:

clientimgname=debian4eWANclient

image files If the images are not found on the kaenvironment tool, the script tries to register it. The installator needs the image file names and the path where to find it.
format:
Image file name.
example:

serverimgfile=debian4eWANserver-msanchon.dsc

format:
Image file name.
example:

clientimgfile=debian4eWANclient-msanchon.dsc

format:
A complet linux path to the folder containing the image files.
example:

imagepath=/home/lyon/msanchon/images

partition To deploy, a partition is needed to be specified to kadeploy. Actually the most current partition is hda6. The partition name to deploy is shown on the message of the day (entering to the site), but no format is defined and the script can't identify the exact part of the message to extract the partition.
format:
Partition name.
example:

partition=hda6

EWAN options

The eWAN options are options from the eWAN graphical interface. This options are specified here to make the config.php file and these are options which the installation programm can't discover by itself.

source deploy option If we don't want to do a full eWAN deployment, we can specify "no" to not deploy the eWAN graphical interface code on the eWANserver node.
format:
yes or no. without quotation marks
example:

ewansourcedeploy=yes

eWAN source file We will specify the file with the eWAN graphical part code.
format:
The file must be a .tar.bz2 file (it will be decompressed).
example:

ewansourcefile=eWANv2.0.tar.bz2

ssh timeout Timeout on each ssh connection
format:
Number of seconds.
example:

sshtimeout=3

minimal CPUs Minimal cpu utilisation for a router, access point and a client to choose a node to emulate.
format:
Number.
example:

cpurout=300

format:
Number.
example:

cpupt_acc=200

format:
Number.
example:

cpucli=100

Tocken buffer parameter Tocken_buffer parameter (netem parameter, used on the link limitation)
format:
Number.
example:

tocken_buffer=15400000

switch Only used on the define logical topology tool
format:
0 or 1
example:

switch_routeur=0

Available subnets Subnets available on the cluster (will be used to be the eWAN connection subnets). The first variable (subnetnumber) defines the number of defined networks, and the others are pairs of ip and mask, one pair for each subnet.
format:
Number for subnetnumber IP address for the subnetnumberXip and its mask for subnetnumberXmask. You must define a pair of variables subnetnumberXip and Xmask where X is the number of subnet which you are defining.
example:

subnetnumber=2
subnetnumber1ip=192.168.5.0
subnetnumber1mask=255.255.255.128
subnetnumber2ip=192.168.6.0
subnetnumber2mask=255.255.255.0

2.2 Installation program (general purpose)

Two parts can be used on other installation programs. The two parts are:

1. the eWAN copy tool, which copies, and deploys the graphical part to the eWAN server node and prepare the environment for a good execution.
2. and the read nodes configuration, which reads the nodes properties to generate the physical configuration file (topology discovery).

copy eWAN

Called from an external node (not the eWANserver). Call:

. deployeWANcopysource -f eWANgraphicalpart -n eWANservernode -u usertouse

On the same folder you must have the two files.

files:

1. deployeWANcopysource
2. deployeWANcopysource2

preconditions:

1. the specified eWAN source file must be in the correct format (.tar.bz2) and in the same folder.
2. the second part of the script must be in the same folder

data:

no configuration data is used.

The first part takes the data from argv (file user and node) and it makes copies of all the needed files (code, DBconfiguration and second part) to the objective node and executes the script second part. The second part starts in the eWAN server node, it unpacks the source and puts it in the correct folder. It also prepares the environment (permisions, ssh-key) and it configures the DB.

read nodes configuration

Called from the eWAN server node. Call:

. deployeWANgetdata

On the same folder you must have the two files and a third file named "nodes" which must contain the list of clients from which discover the topology. One name (or IP address) for each line.

files:

1. deployeWANgetdata
2. deployeWANgetdata2

preconditions:

1. the nodes that are being read, must be in a file named "clients"
2. the correct ssh-keys must allow the server-client connections

data:

1. sshtimeout
2. cpurout
3. cpupt_acc
4. cpucli
5. tocken_buffer
6. switch_routeur
7. subnetnumber
8. subnetnumberXip
9. subnetnumberXmask

These files make the config.php file, which contains all the cluster and eWAN configuration. The network control options are set with the eWAN server node options (gateway, network and mask) and a second script is executed on each client node to get client information (IP, processor speed, network cards number and names and the network control card). The rest of the information is read using the eWAN options (listed in "data") to put them on the config.php file.

Chapter 3

Execution

The execution part is done with the eWAN graphical interface. The graphical interface is deployed from the eWAN node server which serves it using an Apache server (the server can change if the eWAN server is changed). Basically, it allows you to deploy logical topologies, but it administers multiple server configurations, loads and shows topologies, help on the topology creation, and more. A user manual is first presented (with a description of the configuration options).

3.1 eWAN graphical interface user manual

The graphical interface allows users to deploy the eWAN network emulator. This part of the document describes each web page, its possibilities and the links to other pages.

3.1.1 The pages

index.php

Figure 3.1.1.

This page is divided into two parts: The first one has the topology options from the logical topology to deploy with eWAN. You can found three options:

1. **Load a topology** (from your local computer). This option is not useful if you are deploying eWAN in a closed cluster that never allows the connection between your computer and the computer of the eWAN server. You can put your logical topology files on the "examples" folder. This folder will be read and all the files inside will be proposed on the second option (choose among these saved)

2. **Choose among these saved**. Here you have a list of all the topologies that we have in the "examples" folder. If you put a topology inside this folder you will be able to choose this topology by selecting it from the list later. No data is exchanged between the local computer and the computer which contains the eWAN server.

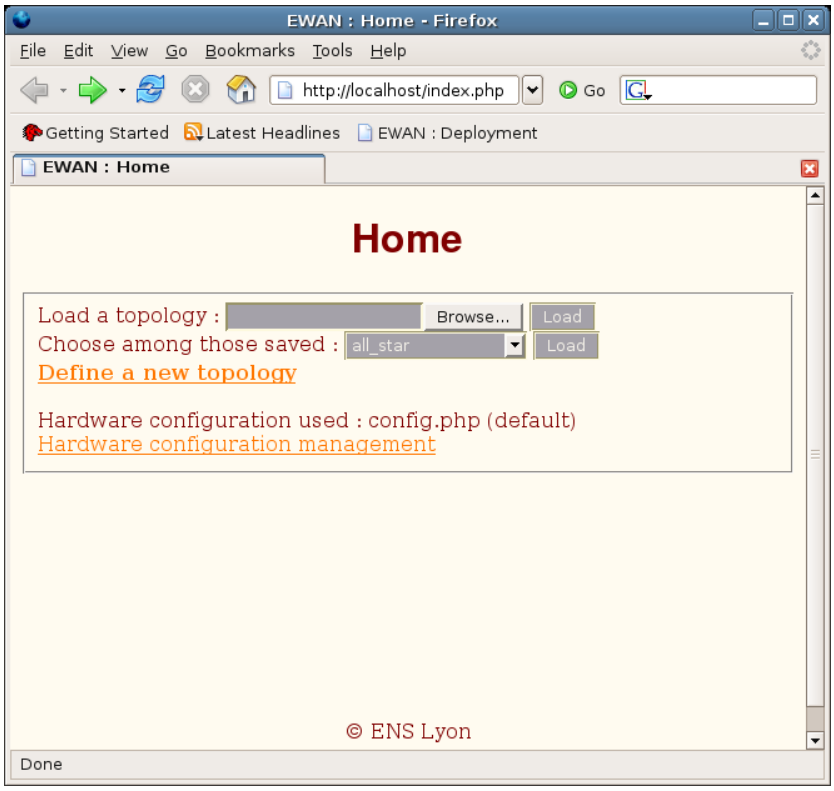

Figure 3.1.1: Initial page

3. **Define a new topology**. You can use this tool to define a basic logical topology, but the tool has a lot of difficulties to save, and it can't be used in an easy, direct way. An example of these difficulties is the inability to use virtual interfaces.

The second one has the configuration options which will lead the physical cluster topology. You know the file from wich all the configuration options are taken, and you have a link to a physical configuration page. If you are using the deployeWAN tool, you do not need to use this part. The data configuration of each physical node is already read by the deploy tool.

charger.php

Figure 3.1.2.

This page loads a topology and shows you the loaded data. Only text definitions are given without a graphical representation, because of the potentially huge amount of data. Graphical representations are optional in this link. Three links are given:

1. **Begin the deployment**. A topology is loaded, you can start the deployment of the logical topology.

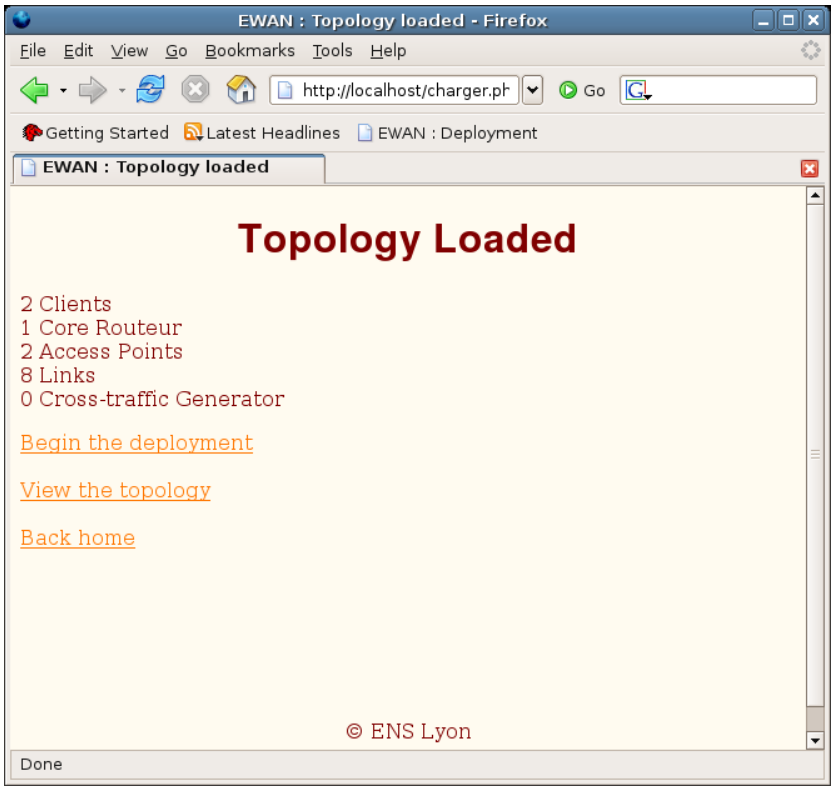

Figure 3.1.2: After load a topology

2. **View the topology**. A graphical representation of the loaded topology allows you to watch the links; the affected nodes by the cross-traffic nodes and the loss, latency, throughput and duplicate properties of each link. This page allows you to have a basic idea of the loaded topology. All the topology properties can be viewed by downloading the topology file (you have the link, on the graphical representation).

3. **Back home**. Return to the index.php page.

materiel.php

Figure 3.1.3.

This page allows the user to choose another physical configuration. If you are using a deployment tool like deployeWAN you have already the good configuration file because the node information is read directly with the deploy tool. Here you have the following options:

1. **Path to the new config file**. Upload a new topology file, but this is not useful if you can't upload files (if cluster not allows external connections). Use the second option if you cannot upload.

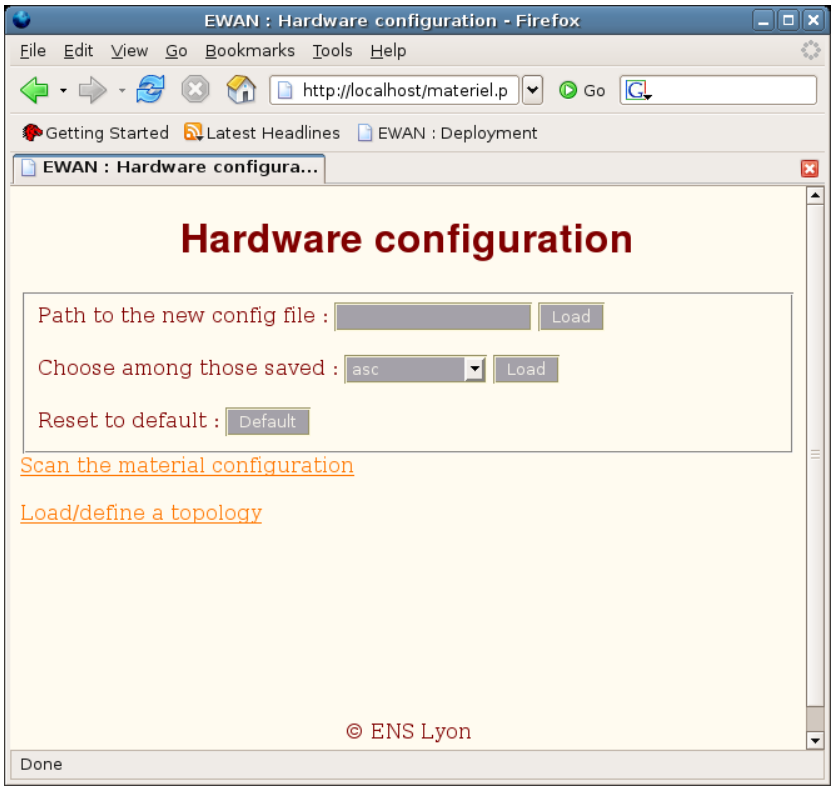

Figure 3.1.3: Physical topologies management

2. **Choose among those saved**. The "configmat" folder is read, and each file is given to make the list that you can see. Choose a topology of the list and push the load button.
3. **Reset to default**. That takes the config.php file as configuration file.
4. **Scan the material configuration**. Scan to verify the physical configuration used.
5. **Load/define a topology**. Push in order to go to the initial page to start defining or loading a logical topology.

interface-emul.php

Figure 3.1.4.

Here you can choose the simulation options of the link. Nowadays, the default link simulation is done using netem, but the changes have been made taking care with the previous versions which uses Nistnet. If you choose netem option, you can deploy topologies with assymetric links. A GNET box can be already used by eWAN. When you validate the options, the information is processed.

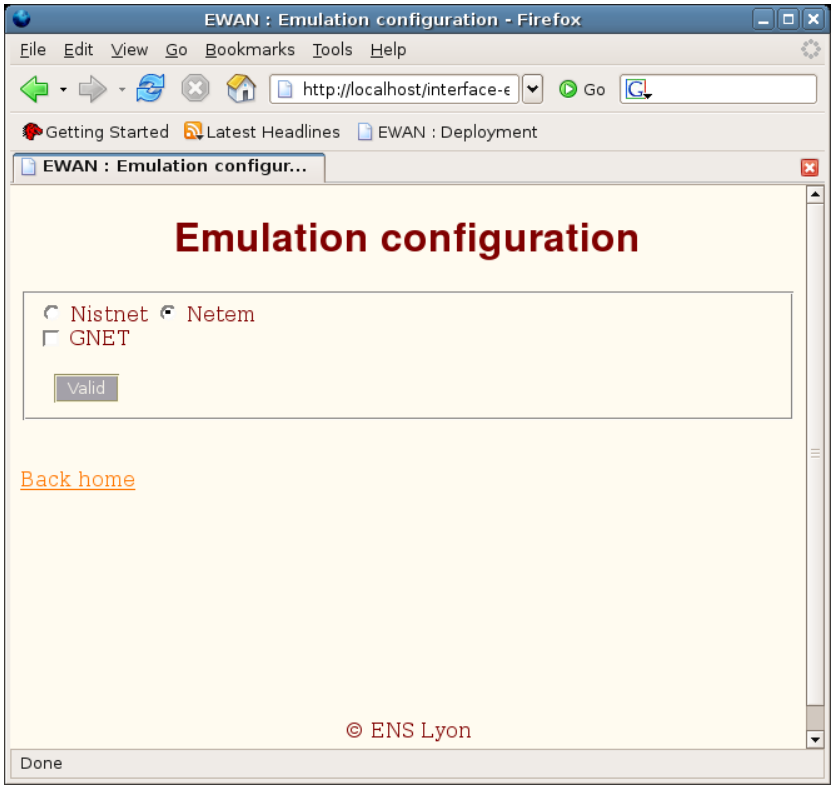

Figure 3.1.4: Link simulation options

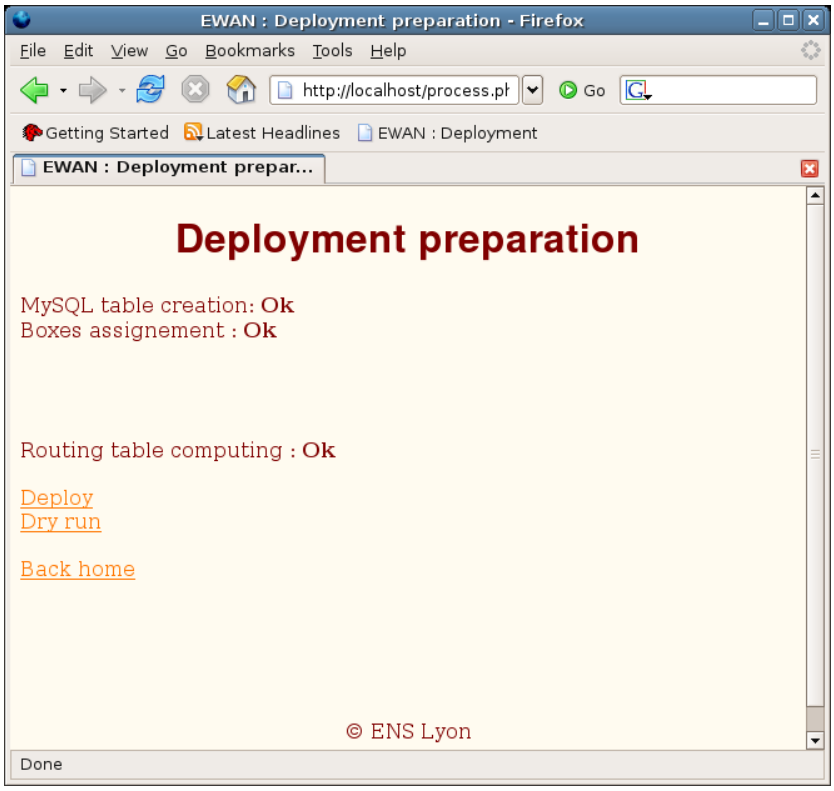

Figure 3.1.5: Topology processed

process.php

Figure 3.1.5.

All the information is processed, the nodes are assigned, the IP networks too, the data is stored on a SQL database, and all the errors, warning chosen options (like the use of virtual interfaces) are shown to you. Finally, three options are available:

1. **Deploy**. Create the scripts for each machine, copy them on the machines and execute the configuration.
2. **Dry run**. Create the scripts for each machine, but do not perform any connection.
3. **Back home**.

deploy.php

Figure 3.1.6.

This page generates the scripts that can be executed on each physical machine to obtain the desired logical topology. Under the "scripts generation"

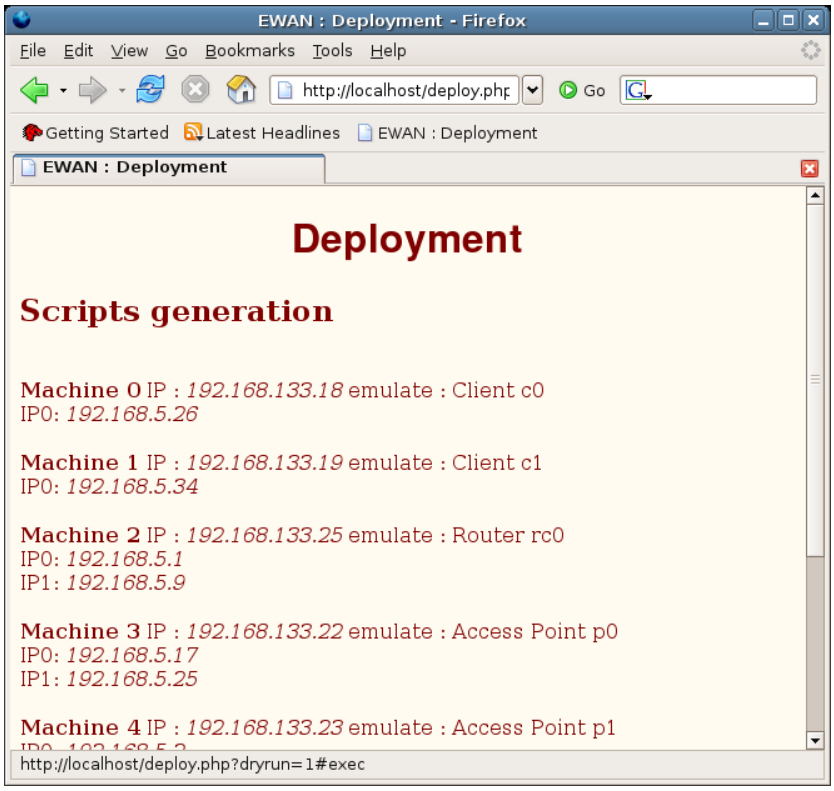
EWAN : Deployment - Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/deploy.php
Go
Getting Started
Latest Headlines
EWAN : Deployment
EWAN : Deployment
Deployment
Scripts generation
Machine 0 IP : 192.168.133.18 emulate : Client c0
IP0: 192.168.5.26
Machine 1 IP : 192.168.133.19 emulate : Client c1
IP0: 192.168.5.34
Machine 2 IP : 192.168.133.25 emulate : Router rc0
IP0: 192.168.5.1
IP1: 192.168.5.9
Machine 3 IP : 192.168.133.22 emulate : Access Point p0
IP0: 192.168.5.17
IP1: 192.168.5.25
Machine 4 IP : 192.168.133.23 emulate : Access Point p1
http://localhost/deploy.php?dryrun=1#exec

Figure 3.1.6: Topology deployment

title, a machine list is printed and with a simple click on the machine name you will get the script for this machine. The page can be called with the dryrun parameter, that causes no script execution, but in a normal deployment with execution the scripts are copied to each client node and executed, all with ssh connections. All the results are printed under the "scripts execution" title and two (or three if dryrun) final links are presented on the page:

1. **Deploy**. If the dryrun option is executed, you can deploy the scripts.
2. **View the topology**. Shows the loaded topology (logical topology) with the IP assignments.
3. **Back home**.

showtopology.php

Figure 3.1.7.

This page allows you to have a fast idea of the logical topology. All the topology elements are printed in a .png image: Each element has its own colour: blue clients, yellow access points, green core routers and red traffic generators for the nodes. The links are in grey if there are no conditions, but they are

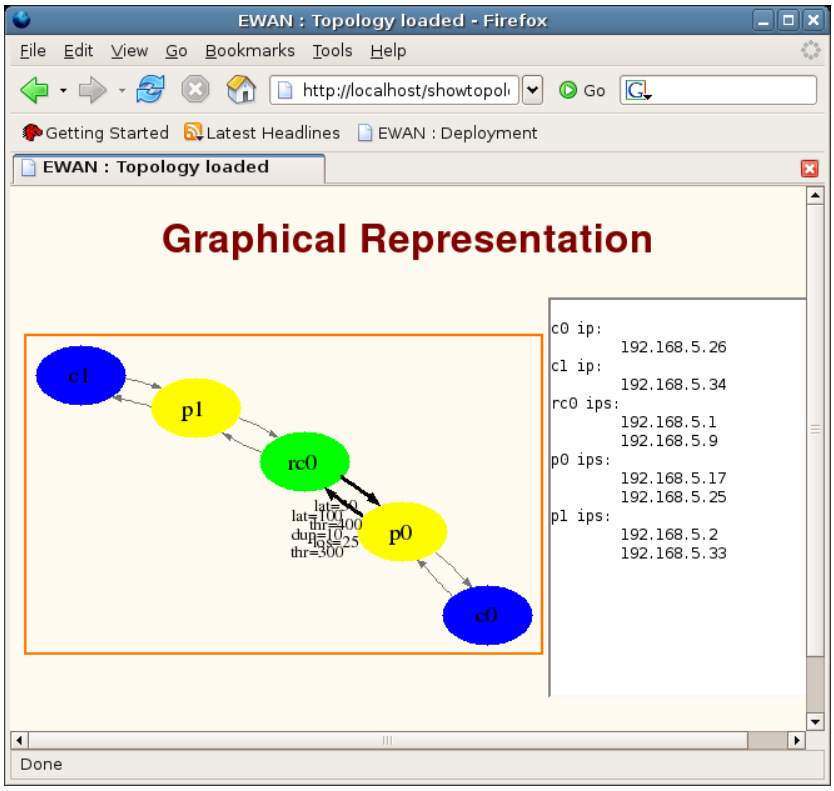

Figure 3.1.7: View the topology

in black and named if they have loss, packet duplicity, throughput limitation or a latency specified. In this case, the link names are their properties. In the cross generators case, the red arrow points to the node which has the interface to be affected to, and the orange arrow points to the node connected to this interface. With these two arrows we can know which interface is affected by the cross traffic generator. If IPs are already specified, a list of nodes and its IPs is shown on the right. Two links (or three if we must deploy) are specified

1. **Begin the deployment**. If you must deploy, the link allows you to begin.
2. **View the exact topology (file)**. Gives access to the topology file on the server. The topology is an XML file and may be your browser that directly reads the xml code, giving you only some data, not the raw .rdf file. Read the source code from the page to get the complete topology file.
3. **Back home**.

verifdispo.php

This page does the verification of your physical topology. It tries to get connection with each client and performs some operations to read the configuration. Yow can keep your old configuration or you can use only the correct machines.

1. **Use only available boxes**.
2. **Keep the old material configuration**. This page is not necessary if you use a deployment tool like deployeWAN for grid5000. The deploy tool performs the correct operations reading the configuration options from each node, and it puts all the information on the default eWAN configuration file (config.php).

interface-topo.php

to define a topology with a graphical interface. This page allows you to define a basic topology in a easy way: access points linked to a simple core. You won't be able to define all the possible topologies with this tool! In each step, the configuration options are looked and respected. Seven steps are applied:

1. **Access points and clients**. Define the number of access points and the number of clients connected to each access point.
2. **Link definition**. The clients with access points links are on a local network. For this reason you will define only the access-point to core routers links. You must choose the farest access point from the core, and then fill the data (latency, loss, duplication). On a second part you must choose another couple of routers and fill the data again. This system is useful because normally, you will know the access point to access point data, and not the access point to core router data.
3. **Throughput limit**. You can specify throughput limits from the local networks on each access point or from the links in the core.
4. **Core topology**. To choose between a star or a ring topology to rely all the core routers.
5. **Cross traffic**. If there are enough machines, a graphical representation of the topology is given and then you can add cross traffic generators by clicking on an interface name.
6. **Other options**. To choose the QoS option, IP version and packet size. These three options are not fully implemented on eWAN nowadays.
7. **End**. To save your configuration file or use it directly on the current session.

Three links are shown on the final page:

1. **Save the configuration file**. Save the topology on your machine, not on the eWAN server.
2. **Begin the deployment**. Deploy with the defined topology
3. **Back home**

3.1.2 The page's links

Structure

- pagename
  - pagesection ⇒ objectivepagename

The links

- index.html
  - Load a topology ⇒ charger.php
  - Choose among those saved ⇒ charger.php
  - Define a new topology ⇒ interface-topo.php
  - Hardware configuration management ⇒ materiel.php
- charger.php
  - Begin the deployment ⇒ interface-emul.php
  - View the topology ⇒ showtopology.php
  - Back home ⇒ index.php
- materiel.php
  - Path to the new config file ⇒ materiel.php
  - Choose among those saved ⇒ materiel.php
  - Reset to default ⇒ materiel.php
  - Scan the material configuration ⇒ verifdispo.php
  - Load/define a topology ⇒ index.php
- interface-emul.php
  - Valid ⇒ process.php
  - Back home ⇒ index.php
- process.php
  - Deploy ⇒ deploy.php
  - Dry run ⇒ deploy.php
  - Back home ⇒ index.php
- deploy.php
  - Execute the scripts ⇒ deploy.php
  - View the topology ⇒ showtopology.php
  - Back home ⇒ index.php
- showtopology.php
  - Begin the deployment ⇒ interface-emul.php

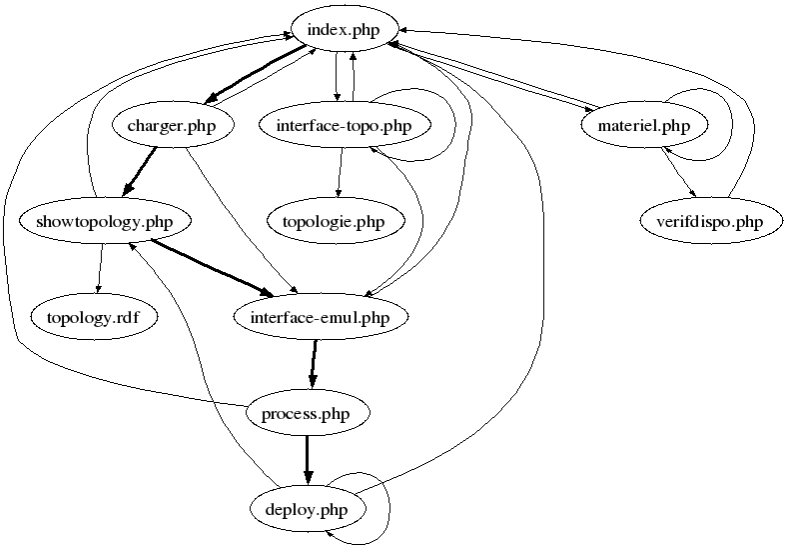

Figure 3.1.8: eWAN graphical interface navigation

  - View exact topology (file) ⇒ topology.rdf
  - Back home ⇒ index.php

- verifdispo.php
  - Use only available boxes ⇒ index.php
  - Keep the old material configuration ⇒ index.php

- interface-topo.php
  - Valid ⇒ interface-topo.php
  - Back ⇒ interface-topo.php
  - Save the configuration file ⇒ topologie.php
  - Begin the deployment ⇒ interface-emul.php
  - Back home ⇒ index.php

3.1.3 Graphical representation

The connections between the GUI pages are represented in the Figure 3.1.8. The bolded path is a path from a normal use case.

3.1.4 Configuration file

The eWAN configuration file (by default config.php) is the only file which contains the eWAN graphical interface options and the cluster definition. If the installation program is used, the file is automatically generated, and the eWAN options must be specified on the deployeWANdata file which contains the installation program configuration options.

If you want to write your own configuration file, you can do it to change different parameters, like the nodes used, IP addresses used by eWAN, minimal cpus, ... EWAN allows you to have several files with distinct configurations and manage it from the graphical interface.

Now a template for the configuration file is written. For each variable you have a little description and if it is automatically generated by the installation tool. Finally you have the list of automatically foud variables, which contains the variables that the installation program could find.

Normal node variables

$machine1=array();

Machine IP on the control network
$machine1['id']='172.24.110.1';

Machine CPU
$machine1['cpu']=1000;

Number of interface to use
$machine1['interfaces']=2;

MTU for normal frames (default 1500)
$machine1['Frames']=1500;

MTU for jumbo frames
$machine1['JumboFrames']=9000;

Machine type: network for routers and links, client for clients and any for routers, links or clients.
$machine1['type']=array('any');

Names of the interfaces used by eWAN. If not defined, eWAN will use eth0, eth1,...
$machine1['if']=array('eth1','eth2');

Define if netem is installed on the machine
$machine1['netem']=true;

Name of the interface control network (used to connect to the control network). By default eth0
$machine1['ctrlif']='eth0';

Array with all the available machines
$machines=array($machine1, $machine2,);

GNET-1 node variables

$gnet1=array();

gnet:<ip of the control machine>:<serial port>:<channel> Channel : 0 : 0/1, 1 : 2/3
$gnet1['id']='gnet:163.220.37.233:16:0';

$gnet1['type']=array('gnet');

Node connected to GNET-1

$gnet1['node']='163.220.37.231';

Interface of this node
$gnet1['ethnode']='eth2';

If you want to use both couples of channels, define 2 gnets variable

Array with all the available machines
$machines=array($machine1, $gnet1,);

Routing options

Ip gw from each control network
$ip_par_defaut=array('172.24.0.0' => ' 172.24.120.10');

Netmask of the control network
$netmask_control='255.255.255.0';

Subnet available on the cluster to be used by the eWAN networks
$temp_ip1=array();

The IP of the subnetwork to use
$temp_ip1['ip']="192.168.5.0";

The mask which defines the subnetwork
$temp_ip1['mask']="192.168.5.0";

Define if the subnetwork is used by default 0, if 1 the subnetwork not will be used
$temp_ip1['full']=0;

Array with all the subnetworks
$ip_privee=array($temp_ip1,);

CPU options

Minimal router CPU
$cpurout = 300;

Minimal access point CPU
$cpupt_acc = 200;

Minimal client CPU
$cpucli = 100;

Other options

Tocken_buffer configuration on the throughput limits
$tocken_buffer=15400000;

Wait comunication seconds for ssh. If negative, eWAN will skip the connectTimeout option
$sshtimeout=3;

Topology creation tool parameter
$switch_routeur=0;

Automatic generation

The next list are the automatic generated variables read by the cluster discover tool.

1. **Machine CPU**
2. **Number of interfaces to use**
3. **MTU for normal frames (default 1500)** (assigned by default)
4. **MTU for jumbo frames** (assigned by default)
5. **Machine type: network for routers and links, client for clients and any for routers, links or clients.** (assigned by default to any)
6. **Names of the interfaces to be used by ewan. If not defined, eWAN will use eth0, eth1,...**
7. **Define if netem is installed on the machine**
8. **Name of the interface control network (used to connect to the control network). By default eth0**
9. **Array with all the available machines**
10. **IP gw from each control network**
11. **Netmask of the control network**
12. **Define if the subnetwork is used by default 0, if 1 the subnetwork not will be used**
13. **Array with all the subnetworks**

Chapter 4

Post work

Once eWAN is deployed, two networks are deployed on the same time. The first is the control network which coincides with the previous network, the cluster network. EWAN gives the access possibility to all the nodes using the control network for the user comfort. With this option, the user can access a node without any gateway, and any machine is the center of all the connections. But this option allows also direct connection between the physical nodes and offers the possibility to work without the deployed topology.

EWAN would force the use of the correct topology and to use it correctly the user must contact the nodes with the correct IP.

The second problem is that the graphical part has a web page with a list of used machines with the emulated node and its ips, and a graphical representation of the topology. But it is not easy to do all the time the nodeIP $\leftrightarrow$ nodeNAME connection.

For these two reasons eWAN will give an automatic translation nodeNAME $\leftrightarrow$ nodeIP in each node of the deployed topology. With this the user can use the names which he knows:

```
ping client032
ssh router2
...
```

Additionally, for a best comprehension, each deployed node has a new hostname, (for example client3) which appears on the prompt when the user connects to the client.

Working with EWAN

Chapter 5

Creating Images for eWAN

On grid5000 you have the kadeploy tool. This tool allows you to deploy environments on the reserved nodes. EWAN needs two different images, one for the eWAN server (which will not be used as a node in your deployed topology) and another one for the clients (routers, clients, access points, emulated links and cross traffic generators in your topology). You can make only one image with all the options needed by the clients and the server if you want but you will do changes on the installation program kadeploy part or write your own installer.

On the next sections you will find what each image must contain and a script with the used commands to create it from a debian4all base image.

5.1 Creating the images

5.1.1 The eWAN server image

This is the image of the eWAN server. Here eWAN is deployed to serve the graphical PHP interface to the user. It has its own private ssh key.

Must contain:

- apache:
  - apache
- php:
  - php4
  - php4-gd
  - php4-mysql
  - php4-domxml
  - php4-cli
- mysql:
  - mysql-client
  - mysql-common

  - mysql-server

- graphs:
  - graphviz

If the eWAN use is done on the own server, the eWAN server image must also contain a web navigator. If the eWAN use is done by a remote connection to the server, then the web navigator must be in the local machine (which access to the eWAN server)

Other conditions:

- SSH keys
  - the private eWAN server key is needed to be included in the folder root.ssh with the name id_dsa
  - the public eWAN key will be put in the authorized keys file, for the extern access to the node without password
- Web server
  - The web server (Apache) must run from the very beginning, listen to the 80 port and have the /var/www/ folder as base data folder
  - On debian, apache2 exists, which replaces apache.
  - Additionally, you can change the initial page from Apache to put some more advisable. eWAN will be installed in the /var/www/eWAN folder to not be merged with other possible Apache projects in the /var/www/ folder.

5.1.2 The eWAN client image

A simple client image to be configured for a correct use like client, router, link emulator or cross traffic generator.
it must contain:

- ssh:
  - ssh
- routing:
  - iproute
- cross-traffic:
  - iperf

Other conditions:

- Ssh keys

  - the public eWAN server key and the public eWAN key have to be put in the authorized keys file to allow the eWAN server connection and eWAN deployment connections without password.

- Optional:
  - traceroute

5.2 After installing the tools

5.2.1 Creating the image

After the script execution and the complete installation of each desired part, you must execute the next command from a node which has acces to the node which has the good image

```
ssh root@node_with_all_the_installed_tools tar --numeric-owner \
--one-file-system -zcSf - / > image_name.tgz
```

5.2.2 Registering in grid5000 (Kadeploy tool)

You don't need to register the image on the kaenvironments tool if you configure the installator tool. For a good configuration, you must have the images on the site to deploy to, the two images on the same folder and a description file for each one. Look on the installator configuration options (2.1.4) to get more information.

Description file example:

```
#debian4eWANserver v1

name = debian4eWANserver
id = 114
version = 1
description = debian for eWAN server
author = magi.sanchon_soler@ens-lyon.fr
filebase = file://home/lyon/msanchon/images/\
debian4eWANserver-msanchon.tgz
filesite = file://home/lyon/msanchon/scripts/\
postinstall/post4all.tgz
size = 1000
initrdpath = /initrd.img
kernelpath = /vmlinuz
kernelparam =
fdisktype = 83
filesystem = ext2
siteid = 1
optsupport = 0
user = deploy
```

5.3 Image creation examples (on grid5000)

Before enter on the node, copy the SSH keys!

5.3.1 Script used to create the server image

```
# packages added
apt-get -y update
apt-get -y upgrade
apt-get -y install apache2
apt-get -y install php4
apt-get -y install php4-gd
apt-get -y install php4-mysql
apt-get -y install php4-domxml
apt-get -y install php4-cli
apt-get -y install mysql-client
apt-get -y install mysql-common
apt-get -y install mysql-server
apt-get -y install graphviz
#apt-get -y install mozilla-firefox
# ssh-keys
# assume that the keys are in the current folder and they are \
named id_dsa_eWAN_server and id_dsa_eWAN.pub
mkdir -p .ssh
mv id_dsa_eWAN_server .ssh/id_dsa
cat id_dsa_eWAN.pub >> authorized_keys2
rm id_dsa_eWAN.pub
# change StrictHostKeyChecking option
cat /etc/ssh/ssh_config | sed \
"s/.*StrictHostKeyChecking.*$/StrictHostKeyChecking no/g" > aux
cp aux /etc/ssh/ssh_config
```

5.3.2 script used to create the client image

```
# packages added
apt-get -y update
apt-get -y upgrade
apt-get -y install ssh
apt-get -y install iproute
apt-get -y install iperf
apt-get -y install traceroute
# ssh-keys
# assume that the keys are in the current folder and they are \
named id_dsa_eWAN.pub and id_dsa_eWAN_server.pub
cat id_dsa_eWAN.pub >> authorized_keys2
cat id_dsa_eWAN_server.pub >> authorized_keys2
rm id_dsa_eWAN.pub
rm id_dsa_eWAN_server.pub
# change StrictHostKeyChecking option
cat /etc/ssh/ssh_config | sed \
"s/.*StrictHostKeyChecking.*$/StrictHostKeyChecking no/g" > aux
cp aux /etc/ssh/ssh_config
```

Chapter 6

eWAN SSH keys (grid5000)

6.1 List of files

- id_dsa
- id_dsa.pub
- id_dsa_eWAN
- id_dsa_eWAN.pub
- id_dsa_eWAN_server
- id_dsa_eWAN_server.pub

Six ssh-key files are used in the deployment of eWAN. The private keys do not have .pub and the public ones have a .pub extension.

6.2 Functions

The used files are grouped on three pairs. Each one with a public and a private ssh-key. Each pair has a function:

6.2.1 id_dsa

The first pair is not provided with the eWAN deployment package because it's a user ssh-key pair.

This pair is used to connect with the portal, site, oar, kadeploy, etc servers on the grid5000 structure 2.1.1.

6.2.2 id_dsa_eWAN

All the deployed nodes (server and clients) will have this public key inside (id_dsa_eWAN.pub) to make it easier to access (no passwords are needed).

The user will have this private part (id_dsa_eWAN), which is used by the installation program.

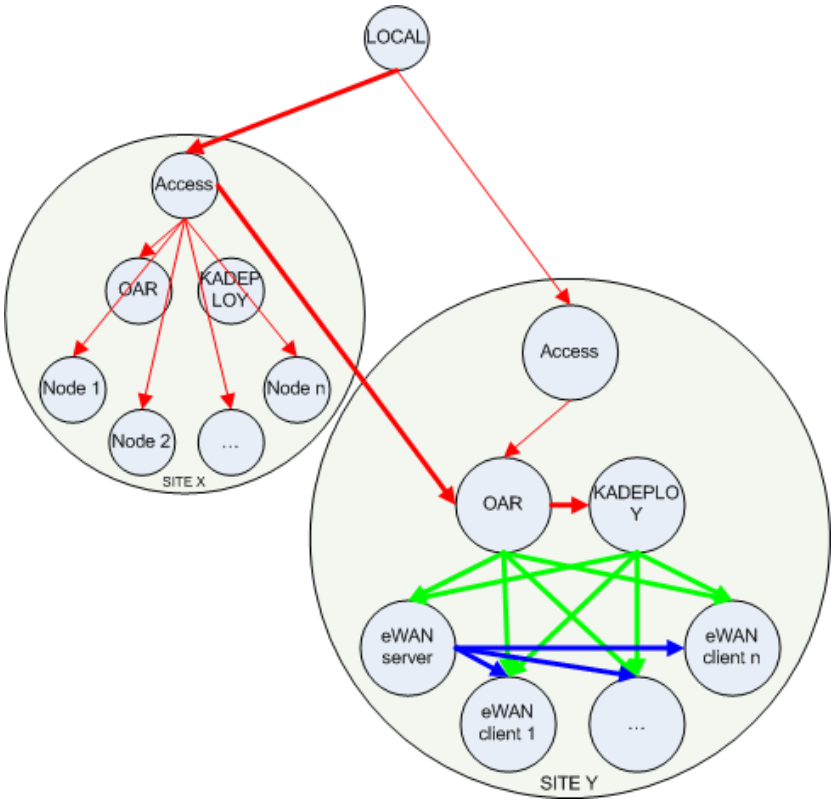

Figure 6.2.1: SSH keys used on eWAN and grid5000. Red=user key, Green=eWAN key and Blue=eWANserver key.

6.2.3 id_dsa_eWAN_server

The last ssh-key pair is for the communication between the eWAN server and the eWAN clients (no passwords are needed). The private key must be placed in /root/.ssh/ folder in the eWAN server, and the public key must be placed in the client's image.

Chapter 7

Defining a topology to deploy

The index page of eWAN allows you to upload a topology to deploy, to use a server side topology (/examples/ folder is read) which is on your machine before the eWAN installation (you can add topologies on the folder) and to define a basic topology with a graphical tool.

The graphical tool allows the definition of basic topologies. Only access points linked with clients and an overdimensioned core (only stars or rings are allowed). If you want define complexe topologies, you must write it manually.

You have a topology RDF schema (which defines exactly the topology options) in the grammar folder. In the next section a fast definition of each part is given with a code example/template.

7.1 The different parts

7.1.1 RDF/XML definition

```
<?xml version="1.0" encoding="iso-8859-15" ?>
<rdf:RDF  xmlns="http://localhost/eWAN/grammar/grammar.php#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" \
    xml:base="http://localhost/" xml:lang="fr">
...
write a "Topology"
...
</rdf:RDF>
```

7.1.2 Topology

Only the name must be defined.

```
  <Topologie rdf:ID="HERE THE TOPOLOGY NAME">
    ...
    write a "Node group"
    and a "Link group"
```

```
  ...
  <QoS>false</QoS>
  <IPv>4</IPv>
  <MTU>Frames</MTU>
</Topologie>
```

7.1.3 Node group

```
<noeuds rdf:parseType="Collection">
...
write Client, Router, Access point or Cross traffic generator
...
</noeuds>
```

7.1.4 Client

Each client is connected to an access point. You must define the client name and the access point name.

```
<Client rdf:ID="HERE THE CLIENT NAME (c...)">
  <ptacces rdf:resource="HERE ACCES POINT NAME (c...)" />
</Client>
```

7.1.5 Router

Only the router name must be defined. The name must begin with the r character.

```
<Routeur rdf:ID="HERE ROUTER NAME (r...)"/>
```

7.1.6 Access point

Only the name must be defined. The name starts with the p character.

```
<Pt-acces rdf:ID="HERE ACCESS POINT NAME (p...)"/>
```

7.1.7 Cross traffic generator

You must define the cross traffic generator name (which starts with the g character), the target router (that is the router which has the interface to attack) and the link which touches the interface and the router.

Only a cross traffic generator for each interface is allowed.

```
<Generateur rdf:ID="HERE GENERATOR NAME (g...)">
  <debit-gen>THOUGHPUT TO GENERATE</debit-gen>
  <lien-cible rdf:resource="#OBJECTIVE LINK"/>
  <noeud-cible rdf:resource="#OBJECTIVE NODE"/>
</Generateur>
```

7.1.8 Link group

```
<liens rdf:parseType="Collection">
...
</liens>
```

7.1.9 Link

In a link you must define the source and the destination nodes (the node names preceded by a # sign), the latency in ms, the loss profile (constant by default), the loss percent (only write the number), the throughput limit in Mb, and the duplication percent.

```
<Lien rdf:ID="l0">
  <from rdf:resource="#HERE THE FROM NODE NAME"/>
  <to rdf:resource="#HERE THE TO NODE NAME"/>
  <latence>LATENCY</latence>
  <perte>
    <Profil-pertes>
      <type-profil>const</type-profil>
      <param-profil>
        <rdf:Seq>
          <rdf:li>LOSS</rdf:li>
        </rdf:Seq>
      </param-profil>
    </Profil-pertes>
  </perte>
  <debit>THROUGHPUT_LIMIT</debit>
  <duplication>DUPLICATION</duplication>
</Lien>
```

7.2 Complete little example

```
<?xml version="1.0" encoding="iso-8859-15" ?>
<rdf:RDF  xmlns="http://localhost/eWAN/grammar/grammar.php#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xml:base="http://localhost/" xml:lang="fr">
  <Topologie rdf:ID="topo">
    <noeuds rdf:parseType="Collection">
      <Client rdf:ID="c0">
        <ptacces rdf:resource="#p0" />
      </Client>
      <Routeur rdf:ID="rc0"/>
      <Pt-acces rdf:ID="p0"/>
      <Generateur rdf:ID="g0">
        <debit-gen>0</debit-gen>
        <lien-cible rdf:resource="#la3"/>
        <noeud-cible rdf:resource="#rc0"/>
      </Generateur>
    </noeuds>
```

```
<liens rdf:parseType="Collection">
  <Lien rdf:ID="l0">
    <from rdf:resource="#c0"/>
    <to rdf:resource="#p0"/>
    <latence>0</latence>
    <perte>
      <Profil-pertes>
        <type-profil>const</type-profil>
        <param-profil>
          <rdf:Seq>
            <rdf:li>0</rdf:li>
          </rdf:Seq>
        </param-profil>
      </Profil-pertes>
    </perte>
    <debit>1000</debit>
    <duplication>0</duplication>
  </Lien>
  <Lien rdf:ID="l1">
    <from rdf:resource="#p0"/>
    <to rdf:resource="#c0"/>
    <latence>0</latence>
    <perte>
      <Profil-pertes>
        <type-profil>const</type-profil>
        <param-profil>
          <rdf:Seq>
            <rdf:li>0</rdf:li>
          </rdf:Seq>
        </param-profil>
      </Profil-pertes>
    </perte>
    <debit>1000</debit>
    <duplication>0</duplication>
  </Lien>
  <Lien rdf:ID="la0">
    <from rdf:resource="#p0"/>
    <to rdf:resource="#rc0"/>
    <latence>0</latence>
    <perte>
      <Profil-pertes>
        <type-profil>const</type-profil>
        <param-profil>
          <rdf:Seq>
            <rdf:li>0</rdf:li>
          </rdf:Seq>
        </param-profil>
      </Profil-pertes>
    </perte>
    <debit>1000</debit>
```

```
        <duplication>0</duplication>
      </Lien>
      <Lien rdf:ID="la3">
        <from rdf:resource="#rc0"/>
        <to rdf:resource="#p0"/>
        <latence>0</latence>
        <perte>
          <Profil-pertes>
            <type-profil>const</type-profil>
            <param-profil>
              <rdf:Seq>
                <rdf:li>0</rdf:li>
              </rdf:Seq>
            </param-profil>
          </Profil-pertes>
        </perte>
        <debit>1000</debit>
        <duplication>0</duplication>
      </Lien>
    </liens>
    <QoS>false</QoS>
    <IPv>4</IPv>
    <MTU>Frames</MTU>
  </Topologie>
</rdf:RDF>
```

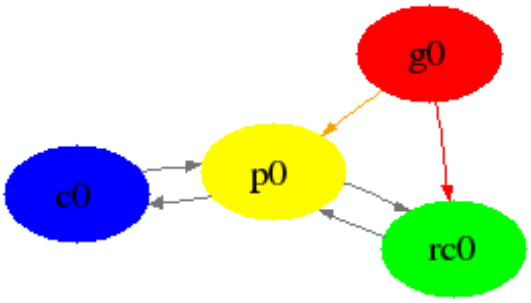

Figure 7.2.1: Topology created

Chapter 8

eWAN archive structure

Once downloaded and uncompressed the eWAN.tar.bz2 file, you will find the next structure.

- **eWANdeploymenttool** has the deployeWAN tool (install eWAN on grid5000)
- **eWANgraphicalinterface** has the eWAN graphical interface which once deployed allow you to deploy your topologies.
- **manual** has this manual. Splitted in parts or the full version.
- **ssh-keys** have the eWAN used keys on grid5000 structure.
- **topology grammar** has the RDF file definition for the topology construction.

Listing

```
eWANv2.0
    COPYING
        deployeWANdata
        eWANdeploymenttool
                deployeWAN
                deployeWANcopysource
                deployeWANcopysource2
                deployeWANdata
                deployeWANgetdata
                deployeWANgetdata2
                deployeWANkadeploy
                deployeWANlocal
                deployeWANoar
                deployeWANportal
                eWANgraphical.tar.bz2
                eWANprova.tar.bz2
                packewandeploy
```

```
eWANgraphicalinterface
        affect.php
        charger.php
        configmat
                asc.php
                config12.php
                config255.php
                config3.php
                gdx.php
                gnet.php
                sallemachine.php
                schedul.php
                template.php

        config.php
        COPYING
        css
                free.css
                mine.css
                site.css
                site.css.old

        deploy.php
        dijkstra.php
        equation.php
        eWANgraphical.tar.bz2
        examples
                all_star.rdf
                all_star_tc.rdf
                asymmetriclink.rdf
                bowtie.rdf
                desequilibre1-nocl.rdf
                desequilibre1.rdf
                desequilibre2-nocl.rdf
                desequilibre2.rdf
                equilibre-nocl.rdf
                equilibre.rdf
                gnet.rdf
                latence.rdf
                locallimit.rdf
                ring_5_complex.rdf
                scheduled-.rdf
                scheduled.rdf
                simple.rdf
                star3.rdf
                star_3_simple.rdf
                star_5_simple.rdf
                star_7_simple.rdf
                star_g0.rdf
```

```
                tc25.rdf
                tc850.rdf
                tc.rdf
                template.rdf
                topologie2.rdf
                topologie.rdf
                tumble.rdf

        grammar
                grammar.php

        img

        inc
                example.php
                footer.php
                headers.php
                head.php
                nav.php
                prologues.php
                var.php

        index.php
        init.sh
        interface-emul.php
        interface-topo.php
        ip.php
        lat2cpu.php
        materiel.php
        packeWANgraphic
        process.php
        routage.php
        showtopology.php
        subnet.php
        topologie.php
        verifdispo.php

manual
        latex
                completeexample.tex
                definetopology.tex
                deployconfigmanual.tex
                deployusermanual.tex
                ewanimages.tex
                ewansshkeys.tex
                eWAN.tex
                graphicalGUIconfigmanual.tex
                graphicalGUImanual.tex
                img
                        assignment.png
```

```
                        charger.png
                        deploypath.png
                        deploy.png
                        emulationoptions.png
                        graphcharger.png
                        graphdeploy.png
                        graphicalinterfacegraph.dot
                        graphicalinterfacegraph.png
                        graphindex.png
                        graphinterfaceemul.png
                        graphmateriel.png
                        graphprocess.png
                        graphshowtopology.png
                        grid5000.png
                        index.png
                        sshkeys.png
                        transformcluster2.png
                        transformcluster.png
                        view.png

        pdf manual parts
                completeexample.pdf
                definetopology.pdf
                deployusermanual.pdf
                ewanimages.pdf
                ewansshkeys.pdf
                filestruct.tex
                graphicalGUImanual.pdf

        eWAN.pdf
prepare
ssh-key
        id_dsa_eWAN
        id_dsa_eWAN.pub
        id_dsa_eWAN_server
        id_dsa_eWAN_server.pub
        README

topology grammar
        grammar.rdf
```

Chapter 9

Complete example

9.1 creating the eWAN images

```
#access to site
magi:~$ grid5000acces -u msanchon -s lyon
msanchon@rork:~$ ssh -A oar

#reservation
msanchon@capricorne:~$ oarsub -q deploy -l nodes=4,walltime=3 -r \
"2006-06-02 12:38:00"
Host:Port = capricorne.lyon.grid5000.fr:59246
 IdJob = 27067
Reservation mode : waiting validation
Reservation valid --> OK

#deploy a basic image
msanchon@capricorne:~$ . grid5000nodesassign 27067
node-13.lyon.grid5000.fr
node-18.lyon.grid5000.fr
node-2.lyon.grid5000.fr
node-6.lyon.grid5000.fr
msanchon@capricorne:~$ . grid5000nodesassign 27067 > actualnodes
msanchon@capricorne:~$ kaenvironments
Checking variable definition...
user selected : msanchon

Name           Version user      Description
----           ------- ----      -----------
debian4all     1 deploy          image debian minimale sans ldap...

msanchon@capricorne:~$ kadeploy -e debian4all -f actualnodes -p hda9

Node    State           Error Description (if any)
----    -----           --------------------------
node-6.lyon.grid5000.fr         deployed
```

```
node-13.lyon.grid5000.fr        deployed
node-2.lyon.grid5000.fr         deployed
node-18.lyon.grid5000.fr        deployed


#copy the ssh-key
magi:~$ scp -r eWANv2.0/ssh-key/ msanchon@acces.lyon.grid5000.fr:
#on the future server
msanchon@capricorne:~$ scp ssh-key/id_dsa_eWAN_server \
root@node-6.lyon.grid5000.fr:
Password:
id_dsa_eWAN_server                  100% 1196     1.2KB/s   00:00
msanchon@capricorne:~$ scp ssh-key/id_dsa_eWAN.pub \
root@node-6.lyon.grid5000.fr:
Password:
id_dsa_eWAN.pub                     100%  589     0.6KB/s   00:00
#on the future client
msanchon@capricorne:~$ scp ssh-key/id_dsa_eWAN.pub \
root@node-13.lyon.grid5000.fr:
Password:
id_dsa_eWAN.pub                     100%  589     0.6KB/s   00:00
msanchon@capricorne:~$ scp ssh-key/id_dsa_eWAN_server.pub \
root@node-13.lyon.grid5000.fr:
id_dsa_eWAN_server      id_dsa_eWAN_server.pub
msanchon@capricorne:~$ scp ssh-key/id_dsa_eWAN_server.pub \
root@node-13.lyon.grid5000.fr:
Password:
id_dsa_eWAN_server.pub              100% 1101     1.1KB/s   00:00

#create the server image
msanchon@capricorne:~$ ssh -A root@node-6.lyon.grid5000.fr
node-6:~# apt-get -y update
node-6:~# apt-get -y upgrade
node-6:~# apt-get -y install apache2
node-6:~# apt-get -y install php4
node-6:~# apt-get -y install php4-gd
node-6:~# apt-get -y install php4-mysql
node-6:~# apt-get -y install php4-domxml
node-6:~# apt-get -y install php4-cli
node-6:~# apt-get -y install mysql-client
node-6:~# apt-get -y install mysql-common
node-6:~# apt-get -y install mysql-server
node-6:~# apt-get -y install graphviz
node-6:~# mkdir -p .ssh
node-6:~# mv id_dsa_eWAN_server .ssh/id_dsa
node-6:~# cat id_dsa_eWAN.pub >> .ssh/authorized_keys2
node-6:~# rm id_dsa_eWAN.pub
node-6:~# cat /etc/ssh/ssh_config | sed "s/.*StrictHostKeyChecking.
*$/StrictHostKeyChecking no/g" > aux
node-6:~# cp aux /etc/ssh/ssh_config
```

```
node-6:~# exit

msanchon@capricorne:~$ ssh root@node-6.lyon.grid5000.fr tar \
--numeric-owner --one-file-system -zcSf - / > \
images/debian4eWANserver.tgz
Password:
tar: Removing leading `/' from member names
tar: Removing leading `/' from hard link targets
tar: /dev/log: socket ignored
tar: /var/run/apache2/cgisock: socket ignored
tar: /var/run/mysqld/mysqld.sock: socket ignored

#create the client image
msanchon@capricorne:~$ ssh -A root@node-13.lyon.grid5000.fr
node-13:~# apt-get -y update
node-13:~#
node-13:~# apt-get -y upgrade
node-13:~# apt-get -y install ssh
node-13:~# apt-get -y install iproute
node-13:~# apt-get -y install iperf
node-13:~# apt-get -y install traceroute
node-13:~# cat id_dsa_eWAN.pub >> .ssh/authorized_keys2
node-13:~# cat id_dsa_eWAN_server.pub >> .ssh/authorized_keys2
node-13:~# rm id_dsa_eWAN.pub
node-13:~# rm id_dsa_eWAN_server.pub
node-13:~# cat /etc/ssh/ssh_config | sed "s/.*StrictHostKeyChecking.
*$/StrictHostKeyChecking no/g" > aux
node-13:~# cp aux /etc/ssh/ssh_config
node-13:~# exit

msanchon@capricorne:~$ ssh root@node-13.lyon.grid5000.fr tar \
--numeric-owner --one-file-system -zcSf - / > \
images/debian4eWANclient.tgz
Password:
tar: Removing leading `/' from member names
tar: Removing leading `/' from hard link targets
tar: /dev/log: socket ignored

msanchon@capricorne:~/images$ ls -l
total 559028
-rw-r--r--  1 msanchon lyon       463 Jun  2 16:26 debian4eWANclient.dsc
-rw-r--r--  1 msanchon lyon 236554240 Jun  2 16:21 debian4eWANclient.tgz
-rw-r--r--  1 msanchon lyon       531 Jun  2 16:26 debian4eWANserver.dsc
-rw-r--r--  1 msanchon lyon 335308800 Jun  2 17:02 debian4eWANserver.tgz
```

9.2 Prepare the environment

To prepare the environment, you must define the .dsc files for each image, and your post install files (the same as the default image, for example). Listed files

below.

9.2.1 The server description

```
#devian4eWANserver v1

name = debian4eWANserver
id = 114
version = 1
description = debian for eWAN server (apache;php4 with gd,mysql\
,dom,xml;mysql with client,common,server;mozilla-firefox)
author = magi.sanchon_soler@ens-lyon.fr
filebase = file://home/lyon/msanchon/images/\
debian4eWANserver.tgz
filesite = file://home/lyon/msanchon/scripts/\
postinstall/post4all.tgz.tgz
size = 1000
initrdpath = /initrd.img
kernelpath = /vmlinuz
kernelparam =
fdisktype = 83
filesystem = ext2
siteid = 1
optsupport = 0
user = deploy
```

9.2.2 The client description

```
#devian4eWANclient v1

name = debian4eWANclient
id = 115
version = 1
description = debian for eWAN client (iproute;iperf)
author = magi.sanchon_soler@ens-lyon.fr
filebase = file://home/lyon/msanchon/images/\
debian4eWANclient.tgz
filesite = file://home/lyon/msanchon/scripts/\
postinstall/post4all.tgz.tgz
size = 1000
initrdpath = /initrd.img
kernelpath = /vmlinuz
kernelparam =
fdisktype = 83
filesystem = ext2
siteid = 1
optsupport = 0
user = deploy
```

9.3 Installation

9.3.1 The configuration file

I don't have a reservation. I must change the deployeWAN (installator) configuration. The final configuration file is:

```
########## Local data ##########
## eWAN ssh-key to use
sshkey=.ssh/id_dsa_eWAN
## grid5000 user
user=msanchon
## sites
# portal site, to connect to
portal=lyon
# deploy site, final site to deploy
deploysite=lyon

########## OAR data ##########
## 0 for a use without previous reservation, the reservation number\
 if the contrary
usereservation=0
## reservation variables
# number of nodes to reserve (server included)
nodes=19
# for how many time in hours take the reservation
time=8
# reserve now for later (reservation for "+ x" hours later). format:\
 + hours
delay=+ 0

########## KADEPLOY data ##########
## image data
# images names (on the kaenvironement tool)
serverimgname=debian4eWANserver
clientimgname=debian4eWANclient
# image file names. real names of the images to register (will be used\
 if the images doesn't exists on the kaenvironments tool)
serverimgfile=debian4eWANserver.dsc
clientimgfile=debian4eWANclient.dsc
# path to the real images
imagepath=/home/lyon/msanchon/images
## image deployment
# deployment partition
partition=hda9

########## EWAN options ##########
## eWAN source deployment options
# choose to deploy ewan server (put its code on the node) or not
ewansourcedeploy=yes
# name of the ewan file to deploy (code file)
```

```
ewansourcefile=eWANgraphical.tar.bz2
## ewan program options (used on eWAN configuration, config.php)
# connection timeout for ssh
sshtimeout=3
## minimal cpu utilisation to choose a node to emulate to
# cpu router
cpurout=300
# cpu acces point
cpupt_acc=200
# cpu client
cpucli=100
# tocken_buffer paramenter (netem parameter, used on the link limitation)
tocken_buffer=15400000
# only used on the define logical topology tool
switch_routeur=0
# subnets available on the cluster (will be used to be the eWAN node \
 connection subnets)
subnetnumber=3
subnetnumber1ip=192.168.5.0
subnetnumber1mask=255.255.255.0
subnetnumber2ip=192.168.6.0
subnetnumber2mask=255.255.255.0
subnetnumber1ip=192.168.7.0
subnetnumber1mask=255.255.255.0
```

To prepare the installation tool, you must execute

```
eWANv2.0$ . prepare
```

9.3.2 Start execution

```
$ . deployeWAN
#########  eWAN deployment  #########
using active ssh-agent...OK
getting variables...OK
adding keys to the ssh-agent:
        own key...OK
        eWAN key...OK
preparing portal script part...OK
connecting to portal...OK
on the deployment site. preparing oar script part...OK
NOT on the oar server. starting the oar part on the oar server...OK
getting variables...OK
taking 19 nodes for 8 hours with a + 0 delay...OK
preparing nodes with job number 27120...OK
preparing kadeploy script part...OK
on the kadeploy server. starting the kadeploy part...OK
getting variables...OK
image register:
        server image(debian4eWANserver) exists
        client image(debian4eWANclient) exists
```

```
getting variables...OK
image deployment:
        server deployment on:
                node-1.lyon.grid5000.fr
                deployment...OK
        client deployment on:
                node-10.lyon.grid5000.fr
                node-13.lyon.grid5000.fr
                node-14.lyon.grid5000.fr
                node-15.lyon.grid5000.fr
                node-16.lyon.grid5000.fr
                node-17.lyon.grid5000.fr
                node-18.lyon.grid5000.fr
                node-19.lyon.grid5000.fr
                node-2.lyon.grid5000.fr
                node-4.lyon.grid5000.fr
                node-5.lyon.grid5000.fr
                node-55.lyon.grid5000.fr
                node-6.lyon.grid5000.fr
                node-7.lyon.grid5000.fr
                node-8.lyon.grid5000.fr
                node-9.lyon.grid5000.fr
        deployment...
                node-10.lyon.grid5000.fr...ERROR
                node-13.lyon.grid5000.fr...OK
                node-14.lyon.grid5000.fr...OK
                node-15.lyon.grid5000.fr...OK
                node-16.lyon.grid5000.fr...OK
                node-17.lyon.grid5000.fr...OK
                node-18.lyon.grid5000.fr...OK
                node-19.lyon.grid5000.fr...OK
                node-2.lyon.grid5000.fr...OK
                node-4.lyon.grid5000.fr...OK
                node-5.lyon.grid5000.fr...OK
                node-55.lyon.grid5000.fr...OK
                node-6.lyon.grid5000.fr...OK
                node-7.lyon.grid5000.fr...OK
                node-8.lyon.grid5000.fr...OK
                node-9.lyon.grid5000.fr...OK
eWAN code deployment:
        getting variables...OK
        deploy eWAN source?...yes
        preparing eWAN script part...OK
        deploy eWAN source.     OK
        copy eWANsource to server...OK
        preparing source-code server script part...OK
        starting source-code server part...OK
        preparing eWAN config script part...OK
        deploy eWAN config.END
connecting to eWAN server:
```

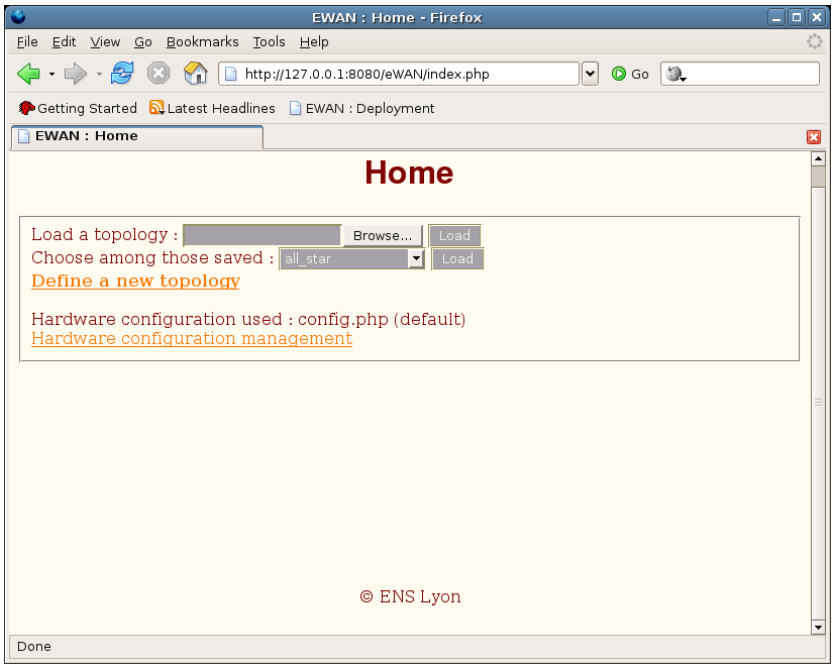

Figure 9.4.1: The first page

```
server deployed. creating connection...OK
open web browser...OK
```

9.4 Execution

The Figures 9.4.1, 9.4.2, 9.4.3, 9.4.4, 9.4.5 and 9.4.6 are a typical use execution.

9.5 Post work

A connection to a node is done by the commands:

```
$ssh -A msanchon@acces.lyon.grid5000.fr
**************************************************************************
For Grid5000 team only.  Unathorized access by any means is forbidden.
Unathorized access may be investigated by any means necessary.
**************************************************************************

 WARNING: It is a criminal offence to:
   - Obtain access to data without permission.
   - Damage, delete, alter or insert data without permission.

---
```

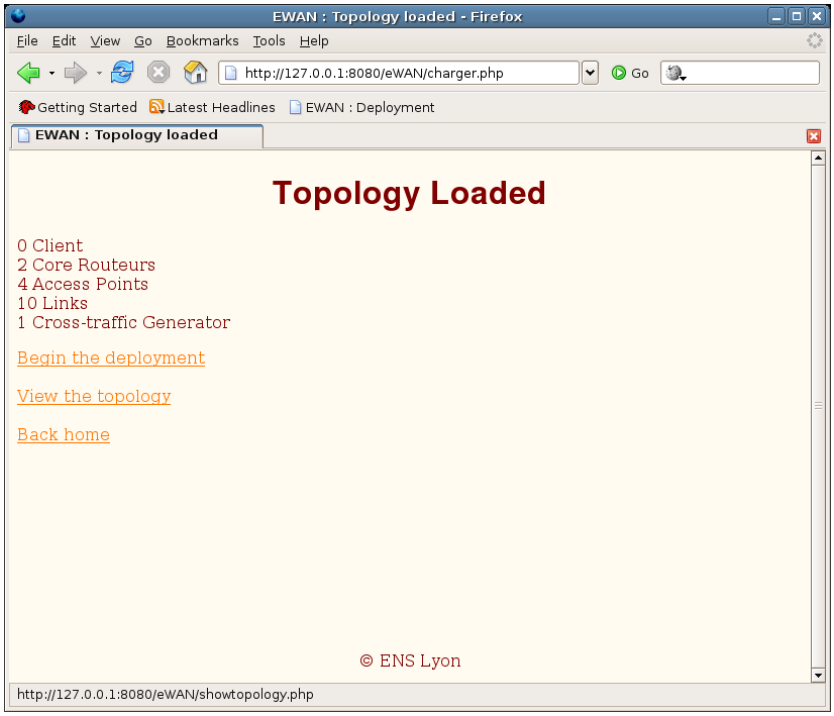

Figure 9.4.2: After charge the desired toplogy

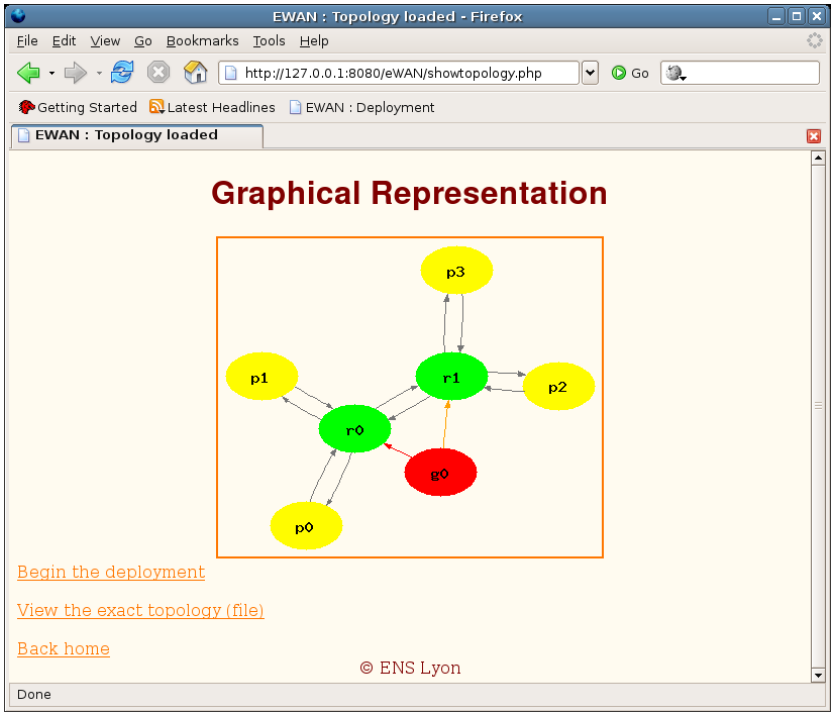

Figure 9.4.3: Graphical represetation of the topology

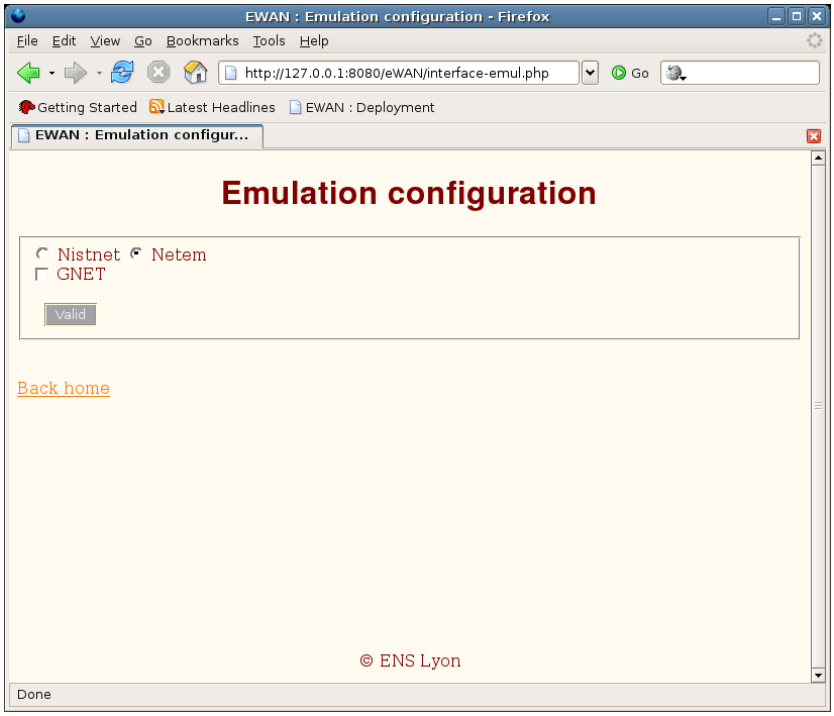

Figure 9.4.4: Link emulation options

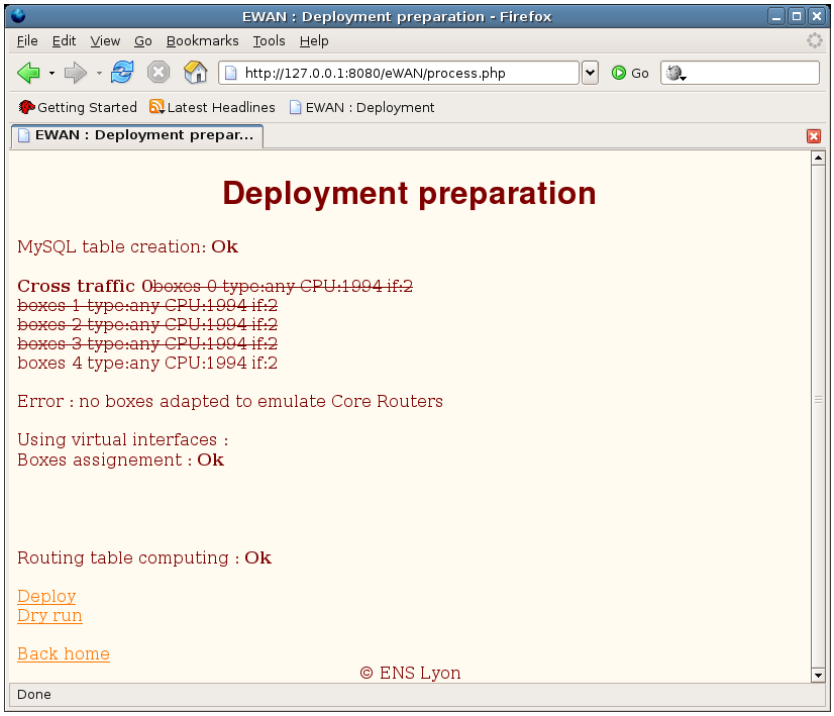

Figure 9.4.5: Assignment resolution

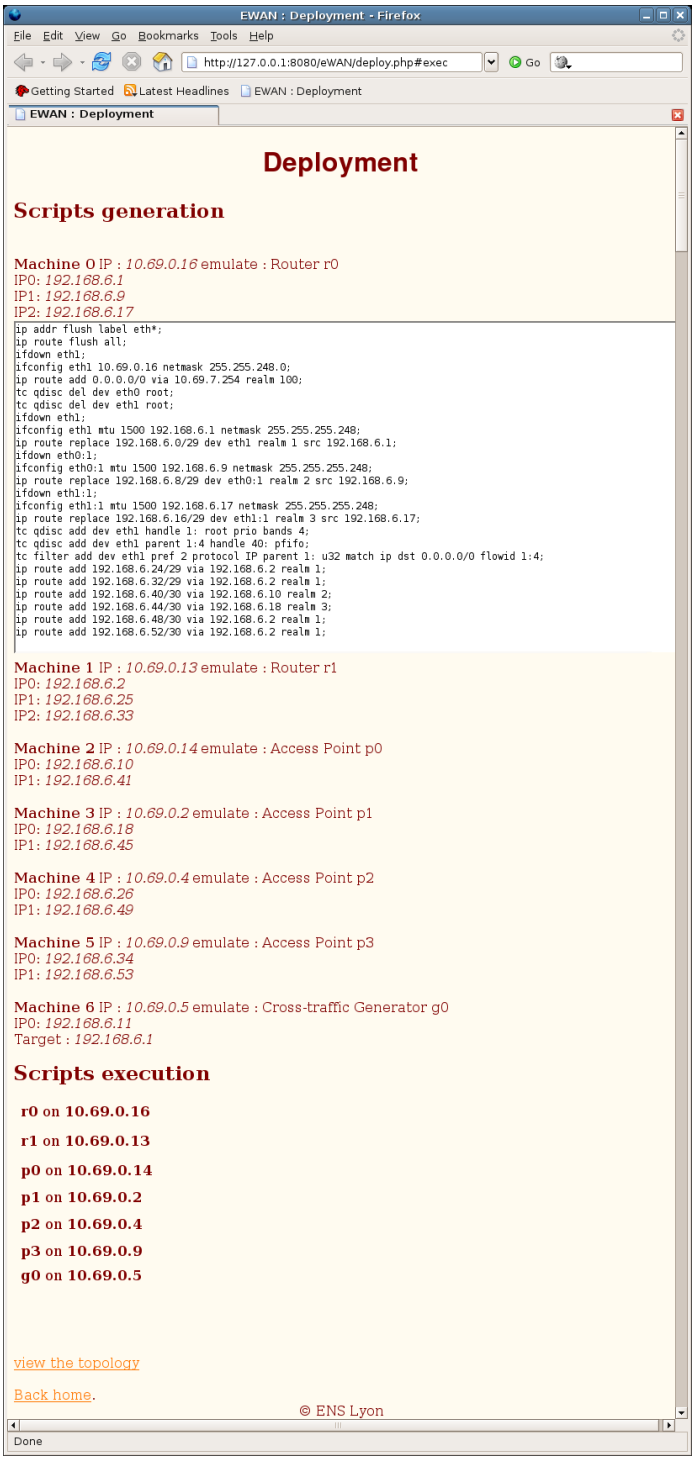
EWAN : Deployment - Firefox
File Edit View Go Bookmarks Tools Help
http://127.0.0.1:8080/eWAN/deploy.php#exec
Go
Getting Started
Latest Headlines
EWAN : Deployment
EWAN : Deployment
Deployment
Scripts generation
Machine 0 IP : 10.69.0.16 emulate : Router r0
IP0: 192.168.6.1
IP1: 192.168.6.9
IP2: 192.168.6.17
ip addr flush label eth*;
ip route flush all;
ifdown eth1;
ifconfig eth1 10.69.0.16 netmask 255.255.248.0;
ip route add 0.0.0.0/0 via 10.69.7.254 realm 100;
tc qdisc del dev eth0 root;
tc qdisc del dev eth1 root;
ifdown eth1;
ifconfig eth1 mtu 1500 192.168.6.1 netmask 255.255.255.248;
ip route replace 192.168.6.0/29 dev eth1 realm 1 src 192.168.6.1;
ifdown eth0:1;
ifconfig eth0:1 mtu 1500 192.168.6.9 netmask 255.255.255.248;
ip route replace 192.168.6.8/29 dev eth0:1 realm 2 src 192.168.6.9;
ifdown eth1:1;
ifconfig eth1:1 mtu 1500 192.168.6.17 netmask 255.255.255.248;
ip route replace 192.168.6.16/29 dev eth1:1 realm 3 src 192.168.6.17;
tc qdisc add dev eth1 handle 1: root prio bands 4;
tc qdisc add dev eth1 parent 1:4 handle 40: pfifo;
tc filter add dev eth1 pref 2 protocol IP parent 1: u32 match ip dst 0.0.0.0/0 flowid 1:4;
ip route add 192.168.6.24/29 via 192.168.6.2 realm 1;
ip route add 192.168.6.32/29 via 192.168.6.2 realm 1;
ip route add 192.168.6.40/30 via 192.168.6.10 realm 2;
ip route add 192.168.6.44/30 via 192.168.6.18 realm 3;
ip route add 192.168.6.48/30 via 192.168.6.2 realm 1;
ip route add 192.168.6.52/30 via 192.168.6.2 realm 1;
Machine 1 IP : 10.69.0.13 emulate : Router r1
IP0: 192.168.6.2
IP1: 192.168.6.25
IP2: 192.168.6.33
Machine 2 IP : 10.69.0.14 emulate : Access Point p0
IP0: 192.168.6.10
IP1: 192.168.6.41
Machine 3 IP : 10.69.0.2 emulate : Access Point p1
IP0: 192.168.6.18
IP1: 192.168.6.45
Machine 4 IP : 10.69.0.4 emulate : Access Point p2
IP0: 192.168.6.26
IP1: 192.168.6.49
Machine 5 IP : 10.69.0.9 emulate : Access Point p3
IP0: 192.168.6.34
IP1: 192.168.6.53
Machine 6 IP : 10.69.0.5 emulate : Cross-traffic Generator g0
IP0: 192.168.6.11
Target : 192.168.6.1
Scripts execution
r0 on 10.69.0.16
r1 on 10.69.0.13
p0 on 10.69.0.14
p1 on 10.69.0.2
p2 on 10.69.0.4
p3 on 10.69.0.9
g0 on 10.69.0.5
view the topology
Back home.
© ENS Lyon
Done

Figure 9.4.6: Deployment information

```
You are on rork, Lyon's Grid5000 portal for resources reservations
You can log on capricorne to access OAR services
Last login: Mon Jun  5 14:20:48 2006 from 140.77.13.90
msanchon@rork:~$ ssh -A root@10.69.0.14
Linux node-1.lyon.grid5000.fr 2.6.12-1-amd64-k8-smp #1 SMP Wed Sep 28 \
02:57:49 CEST 2005 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
p0 :~#
...
```